RECEIVED
CENTRAL FAX CENTER

OCT 25 2006

Section 3, Remarks:

REMARKS

Re-examination and reconsideration of this case is respectfully requested for the 20 claims now in this case in view of the amendments to the claims presented in this Response.

It is noted that the Final Rejection mis-identifies sub-parts and mis-quotes the language of claims 1, 2, 11 and 12. (Portions of the claims quoted in the Final Rejection may be a cut-and-paste copy of a prior Office Action that no longer applies.) Thus, it is difficult for Applicant to fully respond to the Final Rejection. It is not clear to Applicant whether the Office has in fact fully considered Applicant's Responses and Supplemental Responses to the first Office Action of Feb 10, 2006, as the Final Rejection does not refer accurately to the language of the pending claims.

Accordingly, Applicant requests reconsideration and withdrawal of the Final Rejection in view of the language of the claims submitted herein.

Applicant in this Response clarifies the main claims 1 and 11. In particular, Applicant moves former sub part e) ii) of claim 1 into dependent claim 2, sub-part iii), and two of the three options for the object content selection out of sub-part e) iii) into new sub-part x) of claim 2. The corresponding amendments have been made to method claims 11 and 12 to make them consistent.

In view of the apparent mis-understanding by the Examiners of the sticky path functionality, which seems evident from the rejection of claims 8 and 18 over Rochford, Applicant has also made some clarifying amendments to those two claims to emphasize the truly dynamic, automatic, same window, sticky path functionality during scrolling of the hierarchy branches. This dynamic functionality is **not** found in Rochford. Scrolling is a different functionality than point-and-click in Rochford.

The original claims 8 and 18 were, and presently amended claims 8 and 18 are directed to a "during-scrolling" dynamic functionality that has been entirely ignored or overlooked by the Office. That during-scrolling dynamic functionality is clearly described in the Specification as filed and was amply demonstrated during the Webex portion of the Interview. Applicant thought that functionality was understood by the Office; indeed no comment was made that Claims 8 and 18 were in any way unclear. Accordingly, since the

sticky path functionality is still being mis-interpreted, the present amendments more clearly define the dynamic sticky path functionality as originally, and still, claimed. Sticky path is a difference in kind of function as compared to Rochford.

No new matter has been introduced by the amendments to the claims, and no new issues are raised by these amendments, as they are merely clarifying language necessitated by the PTO's withdrawal of the prior rejection and issuing new rejections of claims previously indicated allowable. The amendatory language is follows the previously submitted language of the Original claims as filed and is amply supported in the Specification, including text and drawings.

In order to not burden the file, Applicant incorporates by reference herein the Remarks Section 3 of the Responses filed July 10, 2006 and August 28, 2006. .

Applicant has heretofore made of record the filing on Sept 12 by Express Mail of an archival copy of the three QuickTime demonstration movies 1 – 3: Sticky Path, Classification and Key Phrases functionalities, respectively. These functionalities were demonstrated live in the Webex video conference on Sept 12.

It should be understood that the archival CD QuickTime movies and Webex demonstrations are exemplary only and not the only possible implementations of the features of the inventive claimed MFS system and program. The functionalities are also illustrated in the drawings and Figures as filed, along with the accompanying text of the Specification.

Response to Rejection of Claims 1 – 20:

In spite of what the Final Rejection states, none of **Watkins, Lewak or Rochford,** taken alone or in combination, teach or suggest the claimed system. None of them teach or suggest to one skilled in the art the features in part e of main claim 1 and part c of main method claim 11.

In spite of the quotes in the Office Action out of Applicant's claims, none of the references teach or suggest:

i) key phrase hypertext linking between an object and a collection in which the object is contained, said key phrase comprising at least one of the criteria of said collection;

ii) automatic generation of collection contents by criteria specified for collec-

tion membership through at least one object content attribute selected by user-defined key-phrase matching; or

iii) sticky path hierarchical scrolling

The Office Action notes on page 5 of the Detailed Action significant gaps in **Watkins**. Although those listed are not the only gaps, in order to assist in keeping the prosecution moving forward, Applicant will focus at present on the issue of whether **Lewak** teaches what the PTO claims it teaches. Of course, if it does not then the rejection falls, and all claims are allowable as **Lewak does not cure the defects in Watkins** that the PTO admits by itself (for whatever reason) does not disclose or render obvious the claimed MFS system. Applicant reserves the right to point out other defects in Watkins, including but not limited to that Watkins and Lewak are directed to different functionalities and issues in different programs such that one skilled in the art would not look at the two as being sufficiently similar to look to Lewak as contributing solutions to the defects in Watkins.

Further, the Office Action fails to point out that the alleged motivation it claims to find in Lewak is a reason one skilled in the art would look to Watkins as the base program into which some of the Lewak functionality would be provided. Merely saying OSIA would want a better program is not motivation to combine a feature out of one reference where the sole suggestion to do so is the Applicant's Specification, as Applicant's Specification cannot be the source for the combinational teaching.

Nor does the Office Action provide adequate basis for showing just what gaps in Watkins one skilled in the art would see as needing repair **absent direction in Applicant's Specification,** such that they would look to Lewak, and that the result is the claimed MFS system. That is to say, one skilled in the art can look at Watkins, and depending on mindset and interest find a raft of gaps in Watkins that need repair or filling, and look to Lewak or other references to fill, yet the result is remote from the claimed MFS system.

That points up the fact that **the Office Action still has not purged itself from having improperly used Applicant's Specification as a blue print for picking and choosing only selected functionality out of Lewak to plug into Watkins. That is improper precisely because Applicant's specification is not prior art. That point remains un-rebutted by the Office Action, and for that reason alone, the rejections are unsound and should be withdrawn.**

The Office Action on page 5 relies on Lewak Col 7, line 50 -54 and Col 8, lines 6 – 15 as teaching ***automatic generation of collections*** by selection of user defined key phrase matching, etc, having quoted that language out of main claim 1, sub-part e) iii). (Note the Office action mis-labels that language as out of sub-part e) iv); that language just above is now sub-part e) ii) in main claim 1 and c) ii) in main claim 11). Lewak is not automatic.

In addition, on page 6 of the Final Rejection, the Office also cites Lewak as allegedly teaching, in Col 6, lines 17 – 22, linked categories, and teaching, in Col 5, lines 4 – 31, linking categories assigned to each data file with a reference object by metadata in a File Information Directory.

Now, considering those citations on pages 5 and 6 of the Office Action together, the Office has mis-characterized Lewak's teaching. **That process in Lewak is not automatic. Rather the categories for each file are processed manually by the user. Files are NOT categorized in Lewak AUTOMATICALLY in any way, shape or form.** Further, as noted before (and below), linked categories as described by Lewak, are very different than in the inventive MFS system as claimed.

In this regard, Applicant notes that argument of the Office Action page 5 is a mis-characterization of what Lewak teaches. What Lewak says here is that the "user" **manually** categorizes the files by choosing categories from a limited list in a Categories window. The FC manager, running as a background process, simply checks to see if files that are being closed have already been seen by the system and already categorized "by the user". **If not, the user is asked to categorize the file and the user may (or may not) act in response, and any response needs to be made every time for every file that is closed.**

That is, **Lewak merely provides a simple check and notify function**, no different that the alert one gets in Word, "Do you want to delete this file?". It is up to the user to categorize, and that is done manually. It is not automatic and it is not done by the system using key phrase matching

In essence Lewak's background process is telling the user "You forgot to manually categorize THIS file. Do you want to do so? If so, do it now before the file is really closed, or else it won't get categorized", whether or not the file is in a collection or in any other filing system paradigm. **That is NOT what is claimed, and no overbroad or deliberate misinterpretation by the Office makes it so.**

In the claimed MFS system there is automatic generation of collections, that is, generation is done when an object comes into the computer (or is created therein) without intervention by the user, once the initial configuration is established via user or system defined metadata query specifications, key phrases to match, or combinations thereof. From that point on the claimed MFS system automatically collects objects into various different collections as they are changed either by the system or the user and no notice or reminder or action by the user is required. This is unlike the Lewak system. Quite simply, and contrary to the mis-description in the Office Action, Lewak does NOT teach or suggest any automatic criteria-based or keyphrase-based collections or categorizations.

Indeed, whatever that combination of Lewak and Watkins might be, the PTO can have it, as that is not what is claimed. The PTO can go on manually categorizing each file upon every pop-up file-closing notice if it wants. **The word automatic has meaning in this art, and the Office should acknowledge that and not ignore the patentable significance of such a feature.** Lewak's FC Manager does not do automatic generation of collections, it merely checks to see if the user has in fact previously categorized the particular open file, and lets the user know if he/she has not. What is done, or not done after that is not automatic, it is manual by the user. **What is claimed is not the same and Applicant objects to the Office's deliberate misreading of the claimed subject matter and the overbroad reading of the reference. That is not right, and plays games with the innovation rights of Applicant.**

The Office repeated this point in the rejection of main claim 11. That is the only rejection of main claims 1 and 11. The Office is wrong; the rejection should be withdrawn, and those main claims allowed. Then, as all remaining claims are dependent claims, they are also allowable. Accordingly, Applicant requests allowance of all claims in the case.

However, Applicant makes of record that the Office is also wrong in its position on Lewak in its rejections of other elements of other claims. For example, with reference to claim 2 (and 12), the Office cites Lewak at Col 9, line 56 through Col 10, line 9 as teaching sub-part "iii)" [**this is an example of the Office referring to the wrong set of claims, or having apparently cut-and-pasted the prior rejection language. That language is from claim 2 sub-part ii), NOT sub-part iii)**] the feature of automatically conjoining specifications.

What Lewak does in that description is something very different: creating categories of categories. Here is how Lewak works: Say you want to have a category for each kind of sports

car. You would also then have to define the category Sports Car, which would have as sub-categories Porsche, Mazda, Lotus, etc. Lewak describes, in that cited Column and line, the process of manually selecting the category Sports Car, which then shows only the sub-categories Porsche, Mazda, Lotus, etc, from among which you select the sub-category desired to be viewed.

In contrast, Applicant describes and claims the feature of selecting: first, the Recent Collection, which shows all objects in the "Recent" collection (the collection with criteria "created or modified within the last week"); then the "Photos" collection (the collection with criteria "an image file"), which then shows the objects that are **both** in "Recent" and in "Photos"; then the "Jake" collection (the collection with criteria "my colleague Jake"). The claimed MFS system then conjoins the requests by attribute and automatically shows only those objects that exist commonly **in all three** collections. That is, there may be more Jake images or documents, but may not be photos or recent, or may be photos but not recent. That is not a sub-category structuring, but a conjoining function. To the extent Lewak can be said to refine views by categories, Applicant is refining collections via attribute critera, which is significantly different.

With respect to the Office's argument that Lewak teaches real time filtering in Col 8, lines 16 – 30; Lewak is here just describing ordering the list by name, alphabetically. In patentable contrast, the inventive MFS system employs a number of different and novel filtering functionalities, such as described above for Recent, Images and Jake. Viewing by reference is special in that it shows **only** the RELEVANT categories, once a number of categories have been chosen. In particular, in the MFS system, once a category is selected, ALL other relevant categories are shown, but with their contents filtered by that selection. The Office is commended to review Figures 20 – 22 and accompanying text to better understand the operation of this functionality.

The Office cites Lewak's Col 8, lines 16 – 30 as teaching real time filtering/sorting as claimed on sub-part "x)" [**another example of a mis-cite; should be sub-part vi)**] a Lewak alphabetical display is not real time filtering accompanied by sorting. Filtering is criteria screening, whereas alphabetical display is not a filtering function. Rather, alphabetical display is merely an ordering function, that is, what order to display something without regard for what objects are shown, the latter being a filtering consideration.

. The Office cites Lewak's Col 7, lines 49 – 67 and Col 8, lines 6 – 16 as teaching notification to the user of collection establishment and changes in collections. Here the Office is referring to claim 2, sub-part xi) which is the language pre-Feb 10. The language that should have been examined is in sub-part vi) and reads: "providing a notify event of collection establishment and changes in collections". In the cited columns of Lewak, all he does is notify the user when a file appears in the system that needs to be categorized manually. It is a simple "Hey, do something to this orphan file" function. That is a pre-user-action event. It has nothing to do with automatic notify event of **collection establishment and changes in collections.**

The Office is respectfully requested to read ALL the terms of the claims and the references, and to not over-broadly read the claim (or omit language) in order to make a rejection. That does not meet the mandare of the Commissioner in MPEP 2106, which is directed to the Examining Corps:

> ***"Office personnel should indicate how rejections may be overcome and how problems may be resolved. A failure to follow this approach can lead to unnecessary delays in the prosecution of the application."***

With respect to claim limitations, the over-broad reading approach is NOT approved, the Commissioner stating in the Guidelines:

> ***". . . every limitation in the claim must be considered. Office personnel may not dissect a claimed invention into discrete elements and then evaluate the elements in isolation. Instead, the claim as a whole must be considered. See, e.g., Diamond v Diehr, 450 US at 188 – 89, 209 USPQ at 9 (quoting from the case)."***

Further, in the claimed MFS system, Claim 2, sub-part vi) is directed to the function of sending a notice to the user **after** a collection has a new member due to the **automatic** collection process (run by the system, not the user), or when the collection loses a member due to the automatic collection process. For example, the system can be configured so the user could be notified when a new file that mentions "Apple Computer" is collected into the Apple collection; this can be done completely without user intervention, since another process on the computer may be (say) downloading documents from a central server, which the invention is then automatically organizing into collections. The MFS system is smart enough to recognize an Apple-related document (or image) at the onset of download, and automatically puts it into

the proper collection, then advises the user of what it has done. The MFS system user, unlike the Lewak user, needs do nothing to each incoming file. **That is a patentable distinction not shown by Lewak or Watkins, and nothing in Lewak can morph Watkins into the claimed function, regardless of how the Office attempts to mischaracterize it.**

In connection with part vii) of Claim 2, the Office claims that Lewak teaches Col 8, lines 61 – 67, Col 11, lines 3 – 8 and Col 15, lines 22 – 36 the feature of link creation between objects and collections by various processes. However, in the cited portions of Lewak, he shows: a) given a file the user must manually point-and-click to select a category from a limited menu for the current file: b) selecting more than one category (grouping); c) "linking categories" by which he means that objects in Category A should also be considered to be in Category B. That is, objects in Category "Porsche" are automatically in Category "Sports Car", IF Porsche has been linked to Sports Car. But the claim sub-part is directed to drag-and-drop which Lewak does not show. Nor does Lewak show user entry of collection names, including typing with auto-completion, and does not show automatic linking of categories by the system itself matching meta-data criteria.

In order to not burden the record, Applicant states in summary that the quotes out of Lewak (and occasionally Watkins) with reference to Claims 3, 4, 5, 6, 7, 9, 10, 11, 13, 14,15, 16, 17, 19 and 20 that are cited on pages 7 through 12 of the Detailed Action do not cure the deficiencies in Watkins + Lewak as to the novel and patentable elements e) i) through e) iii) in main claim 1, nor the corresponding steps in main claim 11, steps c) i) through c iii). Further, they do not show the automatic functionality claimed. Accordingly, those claims are patentable, and should be allowed.

As to the rejection of claims 8 and 18, pp 12 – 14 of the Detailed Action, which adds the **Rochford** reference to the other two, that patent is merely directed to a click-and-select, multi-window procedure wherein the single level clicked-on in window 30, 32, 34, 36, 38 in Figs 4A - 7 is reproduced in a separate history-tracking, non-dynamically expandable/ shrinkable (there is no dynamic, automatic resizing) window (finally numbered as 20 in Fig. 7). It does not operate in a true expandable outline view window environment. It simply rewrites a history-of-search list, and shows but a single level in window 30 – 38. It is not automatic during scrolling and does not automatically resize as you scroll up and down a hierarchy.

While Rochford shows a useful program in which a containment hierarchy may be

displayed, and remain visible while sub-items in the branch are scrolled up and down, the characterization of its teaching the claimed sticky path functionality of the MFS system is not correct. The inventive claimed sticky path functionality has the following patentable differences and advantages:

- in Rochford, only the contents in the currently selected branch are visible; in contrast, in the claimed MFS system all the branch contents are visible, as are all items that are siblings, children and, recursively, siblings' children, to the branch that are sorted lower in the display; that is to say, unlike Rochford, in the claimed MFS system, the parent of the items immediately below the sticky path region of the single window is exactly the last item in the sticky path region; the parent of that item is the next above it (the previous item) in the sticky path, and so on; one never sees parents below while scrolling, but rather siblings, their children and in turn their children to the depth of the tree visible in the window, and as more are revealed as you scroll down; keep in mind Rochford is not a fully scrollable system;
- In Rochford, you can only scroll within the current branch contents; in the claimed MFS system you can scroll within the entire tree using the exact same scrolling mechanism that traditional outline displays use;
- The claimed MFS system provides a much simpler interface - **a sticky region** at the top of a single scrolling outline window - and provides **a dynamically updating view** of the currently-visible branch as the user scrolls;
- The claimed MFS system allows the user to determine whether specific branches should be traversed or not during scrolling, by providing a standard mechanism for disclosing selected branches (opening folders or containers in the outline) and not disclosing others. Non-disclosed branches do not stick, and their contents are not traversed. The chart below assists in understanding the patentable distinctions:

Invention	**Rochford**
Extends the outline view with a **"sticky path"** that remains inside the outline frame; the portion of the outline below the path continues to scroll normally. The **sticky path area changes** dynamically during scrolling to indicate the hierarchy above the currently-visible items immediately below the sticky path. Items of all levels are visible below the sticky path, providing context. Clicking on a sticky path item causes the outline to automatically scroll to the beginning of the selected sub-path.	Provides an **external, separate** window or pane that displays the hierarchy above items that are shown in an independent pane. **Moving up in the hierarchy must be done by clicking in the hierarchy pane, rather than simply scrolling.** Scrolling is only permitted within the currently-selected branch of the hierarchy.

The Office Action on page 13 refers to Rochford's Fig 6; Col 2 , lines 62 – 64; Col 3 line 66 through Col 4, line 67; and Col 6, lines 48 -55 as teaching sticky path functionality. Those do not teach that functionality as a fair reading of Rochford clearly shows.

For example, as to Fig. 6 of Rochford, it merely shows in a separate window 38 the contents 70 of a single branch of the hierarchy, which is only a single level (e.g. does not, and cannot, show the nodes below: Barry, Hamilton, Oshawa, etc.) within the same scrolling list 70 in window 38. The list 70 only shows the nodes in the hierarchy at the same level as Barry, etc.

As to Column 3, line 66 through column 4 line 67: this column 4 simply describes a level-by-level navigation method. Rochford is completely unlike the "outline" hierarchy system in Applicant's MFS system in which multiple branches of the hierarchy, and multiple levels, may be displayed as a single list, simultaneously in a continuously scrollable, single dynamic window. In Rochford, only the path to the branch, and the contents of that branch ***at that level***, are visible. Selecting one of those items rewrites it to the path list (44), and replaces the contents of the branch list 70 in window 38 with the contents of the selected (clicked on) branch. At no time can you see the contents of two or more branches simultaneously in window 38.

As to Column 6, line 48-55: this just says that the contents of the single branch ***at one level*** can be scrolled up and down, with a scroll bar or arrow keys. Thus, Rochford is a simple version of a list portion magnifier view, and the hierarchy must be built one level at a time in a separate window, but a window that is not a dynamically, automatically resizing window.

In contrast, the claimed sticky path functionality in claims 1, 11, 8 and 18 is patentably distinct from Rochford, in that it:

- provides a full outline view, showing the contents of multiple branches, at multiple levels simultaneously, allowing the user to scroll up and down to view the contents of each branch;
- provides a view of the path to the items shown at the beginning of the scrollable list (the "sticky path region"), displayed in the same frame as the list.
- automatically resizes both the "sticky path region" and the scrolling region as the user scrolls up and down, navigating through different levels in the hierarchy;
- shows any portion of the hierarchy, at any level, rather than being limited to a single sub-tree and single level of the hierarchy;
- allows the user to quickly scroll back to the beginning of a hierarchy level simply by clicking on the sticky path entry desired; and
- allows the user to quickly scroll back to, and close, a hierarchy level simply by clicking on the disclosure control in the sticky path entry desired.

Further the text of the Application and drawings **as originally filed** clearly addressed and distinguished over the list view of Rochford, pointing out at page 17 "One of the problems with a list view...".

The Examiner is below directed to Applicant's Specification, starting on page 40 (the bold portions below show substantial patentable differences over Rochford). The first paragraphs quoted below describe the operation of an outline view and the paragraphs that follow describe the specific functionality of sticky paths within the context of an outline view environment:

> **"Sticky Paths**
>
> Sticky paths are a unique way of displaying hierarchies of objects within MFS. Often hierarchies of objects are displayed in a sort of **outline view**, whereby objects are listed in some order (typically alphabetical), and **sub-objects that are contained in other objects may be displayed or hidden at the user's control. An object that contains other objects in this way may be either expanded (displaying its sub-objects) or collapsed (hiding them).** Each object has a depth, a numeric value that describes how far down the hierarchy it exists; in particular, how many nodes down the hierarchy tree from the root. Objects at the same depth are known as siblings. The depth determines how far the object is indented to the right in the outline display.
>
> **When an object is collapsed, any object that contains others is indicated in some way with a clickable region, typically a symbol such as a + sign or a triangle, that may be clicked. Clicking on the region expands the object by displaying those objects which are contained within below the object and indented to the right by a specific amount, due to their depth being one greater than the depth of the parent object. Other objects that were at the same level as the object being expanded are moved down the display by the amount needed by the expanded object.**
>
> **Objects within an expanded object may in turn be expanded, resulting in several levels of expanded objects and multiple indentations.**
>
> The ***path*** to an object is defined as the name of the object itself, prefixed by the names of the nested containers in which the object exists in outermost order. For example, if an object E is contained in an object D, and in turn D is contained in C, and C is contained in B, the path to the object E is generally described as B:C:D:E.
>
> In a **highly-hierarchical** display with many objects that do not fit on a single screen, the user must scroll the hierarchy display in order to see objects lower down on the list. **In particular, if some objects have many sub-objects which are in turn expanded to show their respective sub-objects,** it is quite easy to forget what part of the hierarchy one is looking at, i.e., where the user is on the path, since the enclosing objects have scrolled off the top of the display.
>
> **Sticky paths are a mechanism by which a scrollable outline of this form is displayed in two dynamic parts: a path area and a scrollable area. Sticky paths provide the user with a constant awareness of his location in the hierarchy by:**
>
> **1) constantly displaying the current path to the topmost item in the scrollable area above the scrollable area, and dynamically updating the path as**

RECEIVED
CENTRAL FAX CENTER

OCT 25 2006

the objects are scrolled up and down;

2) dynamically resizing the scrollable area to accommodate the path display."

The sticky path functionality operation is supported by reference to Figures 12a, 12b and 28, and accompanying text in Applicant's Specification. One of the movies shown the Examiners and made of record is called The Sticky Path Movie. It not only explains the current state of the art in outline display of hierarchies; demonstrates opening and closing levels of the hierarchy in the outline; defines the concept of the enclosing path to items in a list; shows the limitation of the current state of the art in the Apple Finder; but also then demonstrates the claimed Sticky Path functionality in execution during scrolling; the automatic resizing of the sticky path portion of the window, the disappearing branches as one reverses scrolling to ascend the hierarchy, and fast navigation. Claims 8 and 18 as amended here clearly describe that functionality.

The new rejection of claims 8 and 18 as obvious over Watkins, Lewak and Rochford are in error and should be withdrawn. Rochford also does not cure the defects of Lewak and Watkins as to the functionality and steps in sub-parts e) i) –iii) in main claim 1 and c) i) – iii) in main claim 11, from which these claims 8 and 18 depend. Accordingly the claims are allowable.

CONCLUSION

Favorable action of allowance of all claims is respectfully requested. In the event that there remain any open issues, the Examiner is requested to expedite the prosecution of this case by calling undersigned counsel for Applicant to resolve such issues.

Respectfully submitted,
Dr. Bruce Horn, Applicant

Date: October 25, 2006

by: 
Jacques M. Dulin, Reg. No. 41,067
Attorney for Applicant

Jacques M. Dulin, Esq.
Innovation Law Group, Ltd.
237 North Sequim Avenue
Sequim, WA 98382 3456
e-mail: Dulin@InnovationLaw.com
Phone #: 360-681-7305
Fax #: 360-681-7315

End of Section 3, Remarks **End of Response to Final Rejection**